NISTIR 6401

COMPUTER-INTEGRATED KNOWLEDGE SYSTEM (CIKS) NETWORK: REPORT OF THE 2ND WORKSHOP

Lawrence J. Kaetzel, Editor
K-Systems
Brownsville, MD

Building and Fire Research Laboratory
Gaithersburg, Maryland 20899

United States Department of Commerce
Technology Administration
National Institute of Standards and Technology

NISTIR 6401

COMPUTER-INTEGRATED KNOWLEDGE SYSTEM (CIKS) NETWORK: REPORT OF THE 2ND WORKSHOP

Lawrence J. Kaetzel, Editor
K-Systems
Brownsville, MD

December 1999

Building and Fire Research Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899

Jess and Knowledge Sharing Considerations and Issues

Yannis Labrou and Timothy Finin
University of Maryland, Baltimore County
Department of Computer Science and Electrical Engineering

The main task of this project involved converting a decision tree contained in an engineering decision-support system for bridge coating selection. The system called ***BRCOAT*** (Bridge Coating Expert System) comprised a knowledge base that included multimedia (pictures), which was originally developed in the ***Level5*** expert system development tool. The knowledge base was converted to an equivalent system implemented entirely in ***Jess1***. The larger question is how we can capture the *knowledge* present in such a system in a way that is independent of the particular encoding and implementation. The choice of infrastructure (Level 5 or Jess) is to a large extent independent of the body of domain knowledge, in our case, knowledge regarding paint-coating of bridge structures, that our system expresses. Capturing and expressing this body of knowledge in an implementation-independent manner is a non-trivial problem that has been a long-standing pursuit of the Artificial Intelligence community. Being able to do so would allow us to "transfer" this knowledge to whatever particular implementation is appropriate for the task and circumstances, or, in other words, to make it sharable between applications. Moreover, updating this *knowledge base* would not require a massive overhaul of our implementation(s).

In the type of system we dealt with, knowledge is represented in the form of *rules*. The rules themselves make use of *terms* that have a particular meaning for those familiar with the domain (such as *riveted*, or *bolted*, or *salts*). Thus, our problem has two aspects: (1) expressing the rules in a neutral format, and (2) expressing the meaning of the terms in an application-independant (but probably domain dependent) fashion. The goal of sharing the content of formally represented knowledge has been at the center of a research consortium called the Knowledge Sharing Effort or KSE. The KSE, started in the early 90's, aimed at developing tools and infrastructure for the sharing of knowledge both at design-time and run-time. The two results of the KSE that are most relevant to our goals are KIF (Knowledge Interchange Format) and Ontolingua.

Knowledge Interchange Format (KIF)

Specialized languages have been developed which are particularly good at describing certain fields. For example, STEP (Standard for the Exchange of Product Model Data) is an ISO standards project working towards developing mechanisms for the representation and exchange of a computerized model of a product in a neutral form. SGML is an example of a language, which is designed to describe the logical structure of a document. There are special languages for describing workflow, processes, chemical reactions, etc. It would be nice if there were some expressive languages and related computer systems which were good at representing a very broad range of things, like the natural languages, but which do not suffer the problems of imprecision and ambiguity. Database systems and their languages (e.g., SQL, OQL) offer one general approach and certain object-oriented languages perhaps offer another. However, it is difficult or impossible to capture all kinds of information and knowledge in most of these general languages.

KIF is a particular logic language and was proposed by the KSE as a standard to describe things within computer systems, e.g. expert systems, databases, intelligent agents, knowledge bases, *etc*. Moreover, it was specifically designed to make it useful as an "interlingua". By this we mean a language which is useful as a mediator in the translation of other languages. For example, people have built translation programs that can map a STEP/PDES expression into an equivalent KIF expression and vice versa. If we had another language for electronic commerce, say MSEC, we could develop a translator to and from KIF. We would then have a way to translate between STEP/PDES and MSEC using KIF as an intermediate representation.

So, what are the issues regarding translation between Jess (or CLIPS) and KIF? In general, providing a complete translation from one language (Jess) to another (KIF) and vice versa is a hard or even intractable problem because the logics expressed by the two languages might not be equivalent. The experience of the Stanford researchers involved with KIF, suggests that it is very unlikely for one to get a general-purpose translator from KIF into CLIPS. Often though, for practical applications we do not need a complete translation; translating a sufficiently expressive subset of the source language to the target language, could be enough for many knowledge bases. We further discuss this notion of a partial translation when discussing Ontolingua. We are not aware of any results investigating the other direction of the translation, *i.e.*, from CLIPS (or Jess) into KIF.

Ontolingua

The question of the meaning of the terms is a more difficult one. Knowledge bases are designed with a particular view of the world; they have different models of the world in which objects, classes and properties of objects of the world may be conceptualized differently. For example, the same object (in the real world) may be named differently or the same term may have different definitions. In addition, knowledge bases may conceptualize different taxonomies from different perspectives. Therefore, to ensure accurate sharing of knowledge between any two knowledge bases they must agree on the model of the world, at least the part of the world about which they are exchanging information with each other, or to use the proper terminology, they must share a common ontology. An ontology for a domain is a conceptualization of the world (objects, qualities, distinctions and relationships, etc. in that domain). Such a specification should be an objective (i.e., interpretable outside of the knowledge base) description of the concepts and relationships that the applications use and refer to. A shared ontology can be in the form of a document or a set of machine interpretable specifications.

As a concrete representation in a computer, an ontology consists of a specification of concepts to be used for expressing knowledge including the types and classes of entities, the kinds of attributes and properties they can have, the relationships and functions they can participate in and constraints that hold over them. Ontologies are distinguished from knowledge-bases in general not by their form, but by the role they play in representing knowledge. Ontologies are typically consensus models for a particular domain. They place an emphasis on properties that hold in all situations rather than on one or more particular ones. Consequently, their implementations put an emphasis on representing generic classes over specific instances. Ontologies do not tend to change over the course of their use in problem solving, so are well suited to compiling into programming tools and environments. Since ontologies by their nature need to support and satisfy a community of users (and developers) there is a emphasis on collaborative development and on the ability to translate them into multiple representational formalisms.

Within the context of the KSE, researchers at Stanford’s Knowledge Systems Laboratory have developed a set of tools and services to support the process of achieving consensus on common shared ontologies by geographically distributed groups. These tools are built around Ontolingua, a language designed for describing ontologies with it, and make use of the world-wide web to enable wide access and provide users with the ability to publish, browse, create, and edit ontologies stored on an ontology server. Users can quickly assemble a new ontology from a library of existing modules, extend the result with new definitions and constraints, check for logical consistency, and publish the result back to the library. In addition, the KSL Ontology server is able to translate Ontolingua ontologies into a number of other representation formalisms such as LOOM, CLASSIC, KIF and CLIPS. Actually, Ontolingua has two different translations into CLIPS; translating to and from

Ontolingua is for the most part equivalent to translating to and from KIF since Ontolingua is designed on top of KIF. One translation tries to translate into the OO substrate of CLIPS; this translation is not too good, and is limited largely by the representational inadequacies of CLIPS's OO system. The other translates the KIF sentences into a direct CLIPS analogue of the KIF sentences. The conclusion is that neither of the two methods provides a foolproof translation.

Knowledge Query and Manipulation Language (KQML)

Besides KIF and Ontolingua there was a third crucial component in the KSE approach, the KQML language (Knowledge Query and Manipulation Language). KQML is a high-level language and protocol for applications (such as knowledge bases or information agents) that need to exchange knowledge. This exchange is a level above the process of transporting bits and bytes; it involves higher-level informational attitudes about knowledge, such as querying, informing, expression of interest in, ability to deliver information about, and so on. The idea was for KIF, Ontolingua and KQML to be used in unison: the represented knowledge would be expressed in KIF (or translated to and from KIF), the various terms would have meaning captured in ontologies defined in Ontolingua and KQML would provide the higher-level messaging protocol for real-time interaction between applications. Of course KIF, Ontolingua and KQML can be used independently of each other in order to solve different aspects of the knowledge-sharing problem. So, it does not matter to KQML whether applications exchange knowledge represented in Jess (or pure CLIPS for that matter) or KIF.

Knowledge Sharing and the Web

The KSE started at a time when the WWW was not a part of everyday vocabulary. An important issue is how to make use of these languages and technologies in a web environment. A recent important development is the emergence of *XML* (eXtended Markup Language). XML is an SGML-type language and format that is often promulgated as the successor to HTML. Technically speaking, XML itself is not a single markup language but a meta-language that lets users design their own markup language. A regular markup language defines a way to describe information in a certain class of documents (*e.g.* HTML). XML lets you define your own customized markup languages for many classes of document. Another effort that is of interest is RDF (Resource Definition Framework). RDF is designed to provide an infrastructure to support metadata across many web-based activities and is intended to provide a uniform and interoperable means to exchange the metadata between programs and across the Web. Furthermore, RDF will provide a means for publishing both a human-readable and a machine-understandable definition of the property set itself. RDF will use XML as the transfer syntax in order to leverage other tools and code bases being built around XML.

XML and possibly RDF will allow for ontology definitions that are instantly web-ready and web-accessible. Does this make Ontolingua obsolete? Ontolingua is an entire language and set of tools tailored for ontology development; what will probably happen (actually Stanford KSL researchers are already exploring this direction) is a translation of Ontolingua specifications into XML. This would make XML the format of choice for encoding and expressing ontologies. But the features of XML offer other possibilities, too: the authors are currently working at UMBC on XML encodings for KIF and KQML and other researchers are experimenting with XML encodings of Jess rules. The ultimate goal is to use XML as a universal encoding for rules, term definitions and message exchanges and to remove the barrier of what the human user understands and what the machine interprets. At least in theory, XML offers the promise of a continuum where knowledge is dynamically captured and accessed by both human and machine.

Conclusion

Further work on this project should include knowledge sharing as a primary goal. It should also focus on XML and the possibilities it offers for expressing the implicit ontology that the knowledge base encompasses. Also, an XML-encoding of KIF rules would be an interesting way to represent the knowledge captured in the Jess rules themselves. Given the lack of tools (and experience) for translation from Jess (or CLIPS) into KIF, we would rather re-write the knowledge base in KIF than attempt to translate the Jess rules into KIF rules. A major problem with such a re-write would be the graphical interface that in the Jess version of the system is defined as Jess rules. But the appeal of XML is that it would allow us to separate the domain knowledge and the problem-solving logic (the rules that actually encode the domain knowledge) from the *presentation,* which is going to be delivered via XML and a browser.